In the Claims

1. (Currently amended) A computing device comprising:

a processor programmed with a client that ~~can~~ operates with a parser or generator for both text and binary mark up languages~~; in which~~ , whereby:

the client uses a unique integer value ~~that can be interpreted~~ interpretable in an index of elements, attributes and attribute values needed to describe a particular type of mark-up document, the index mapping ~~that~~ the unique integer value to:

(a) [[to]] a token associated with ~~prededefined~~ a predefined element, attribute or attribute value to enable a token based mark up language to be handled; and ~~also~~

(b) [[to]] a string associated with a ~~prededefined~~ predefined element, attribute or attribute value to enable a string based mark up language to be handled.

2. (Original) The device of Claim 1 in which the text mark up language is XML and the binary mark up language is WBXML.

3. (Previously presented) The device of Claim 1 in which a table of mappings of each of the tokens to each of the strings is created and each mapping is given one of the unique integer values.

4. (Original) The device of Claim 3 in which two lists of unique integer values are created: one indexed on tokens and the other indexed on the index of the position of a string in a string pool table.

5. (Previously presented) The device of Claim 1 in which there is an extensible framework that accepts one or more mark-up language parsers and/or generators, each implemented as plug-ins to the framework, with different plug-ins enabling different kinds of mark up languages to be handled by the device.

6. (Original) The device of Claim 5 in which there is a namespace plug-in to the extensible framework that sets-up all the elements, attributes and attribute values for a namespace.

7. (Original) The device of Claim 6 in which the index is encapsulated in the namespace plug-in and therefore is insulated from the client, parser and generator.

8. (Currently amended) A method of parsing a mark-up language document on a computing device, comprising:

~~the step of~~ configuring a client ~~using~~ on the computing device to use a unique integer value that is interpreted in an index of elements, attributes and attribute values needed to describe a particular type of mark-up document, the index mapping that unique integer value to:

(a) [[to]] a token associated with ~~prededefined~~ a predefined element, attribute or attribute value to enable a token based mark up language to be handled; and ~~also~~

(b) [[to]] a string associated with a ~~prededefined~~ predefined element, attribute or attribute value to enable a string based mark up language to be ~~handled~~ parsed.

9. (Currently amended) A method of generating a mark-up language document on a computing device, comprising

~~the step of~~ configuring a client ~~using~~ on the computing device to use a unique integer value that is interpreted in an index of elements, attributes and attribute values needed to describe a particular type of mark-up document, the index mapping that unique integer value to:

(a) [[to]] a token associated with ~~prededefined~~ a predefined element, attribute or attribute value to enable a token based mark up language to be handled; and ~~also~~

(b) [[to]] a string associated with a ~~prededefined~~ predefined element, attribute or attribute value to enable a string based mark up language to be ~~handled~~ generated.

10. (Previously presented) The method of Claim 8 in which the text mark up language is XML and the binary mark up language is WBXML.

11. (Previously presented) The method of preceding Claim 8 in which a table of mappings of each of the tokens to each of the strings is created and each mapping is given one of the unique integer values.

12. (Previously presented) The method of preceding Claim 8 in which two lists of unique integer values are created: one indexed on tokens and the other indexed on the index of the position of a string in a string table.

13. (Previously presented) The method of preceding claim 8 in which there is an extensible framework that accepts one or more mark-up language parsers and/or generators, each implemented as plug-ins to the framework, with different plug-ins enabling different kinds of mark up languages to be handled by the device.

14. (Original) The method of Claim 13 in which there is a namespace plug-in to the extensible framework that sets-up all the elements, attributes and attribute values for a namespace.

15. (Original) The method of Claim 14 in which the index is encapsulated in the namespace plug-in and therefore is insulated from the client, parser and generator.